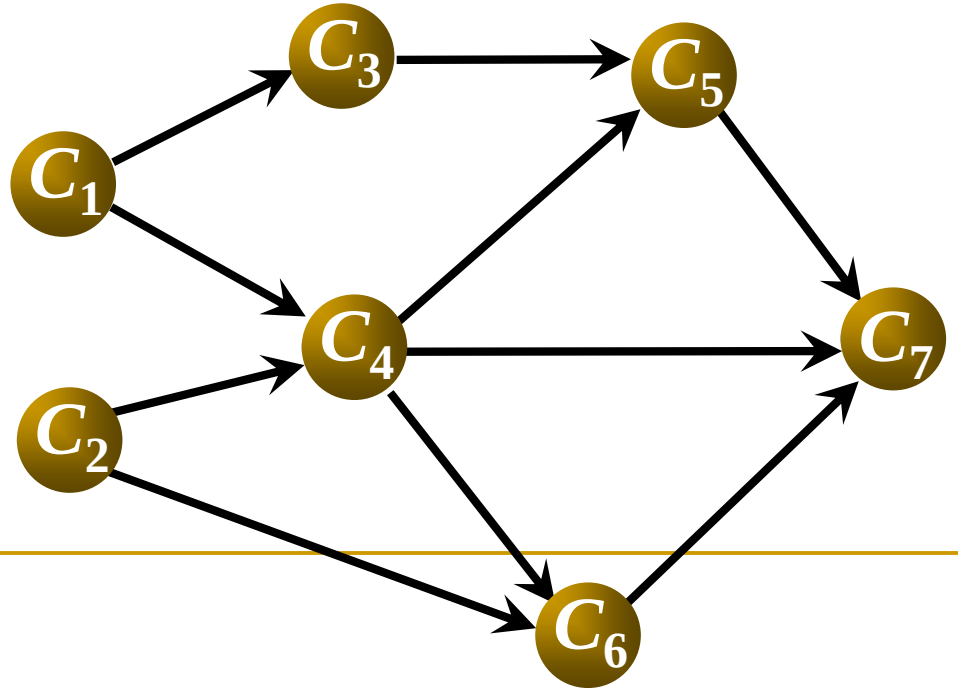
Topological Sorting

Bruce Nan

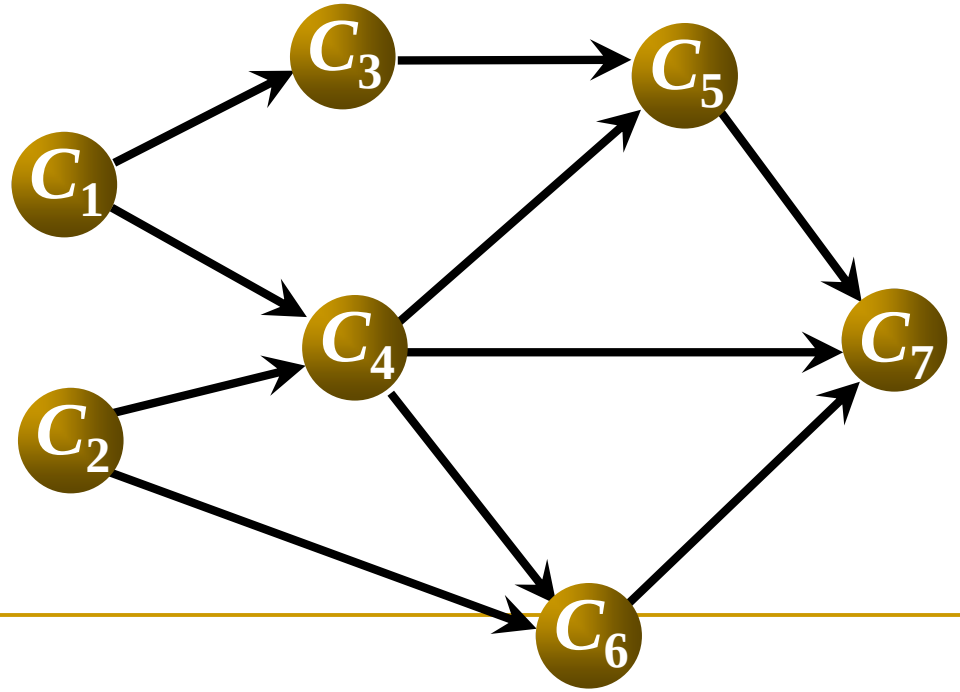
Topological Sorting

- A topological sort is an ordering of vertices in a directed acyclic graph, such that if there is a path from vi to vj, then vj appears after vi in the ordering.



Topological Sort Example

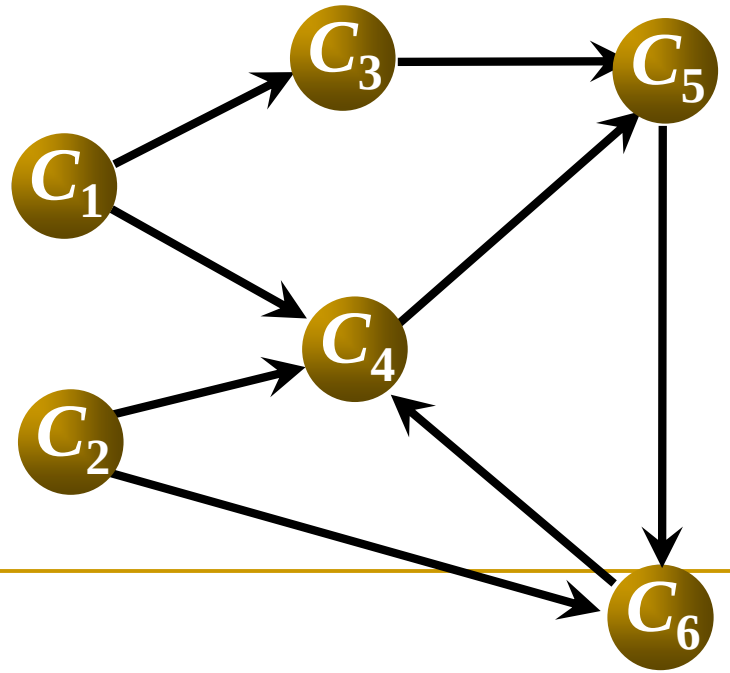
- The following graph represents the course prerequisite structure at a state university in Miami.
- A directed edge (v,w) indicates that course v must be completed before course w may be attempted.



A topological ordering of these courses is any course sequence that does not violate the prerequisite requirement.

Notice

- It is clear that a topological ordering is not possible if the graph has a cycle, since for two vertices v and w on the cycle, v precedes w and w precedes v.
- Furthermore, the ordering is not necessarily unique; any legal ordering will do.



How to realize Topological Sort?

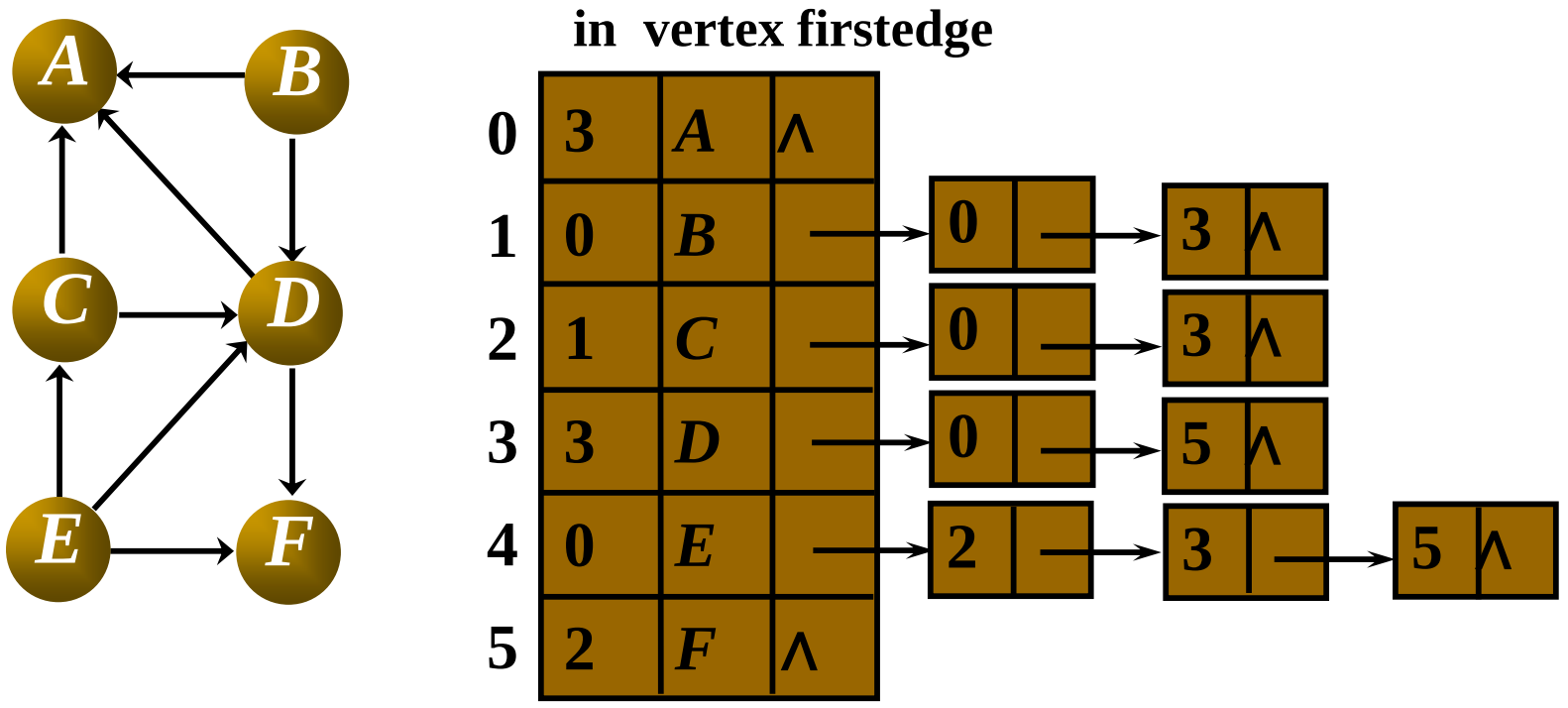
- Basic Idea:
  - A simple algorithm to find a topological ordering is first to find any vertex with no incoming edges.
  - Then, we can print this vertex, and remove it, along with its edges, from the graph.
  - We apply this same strategy to the rest of the graph until all vertices have been output.

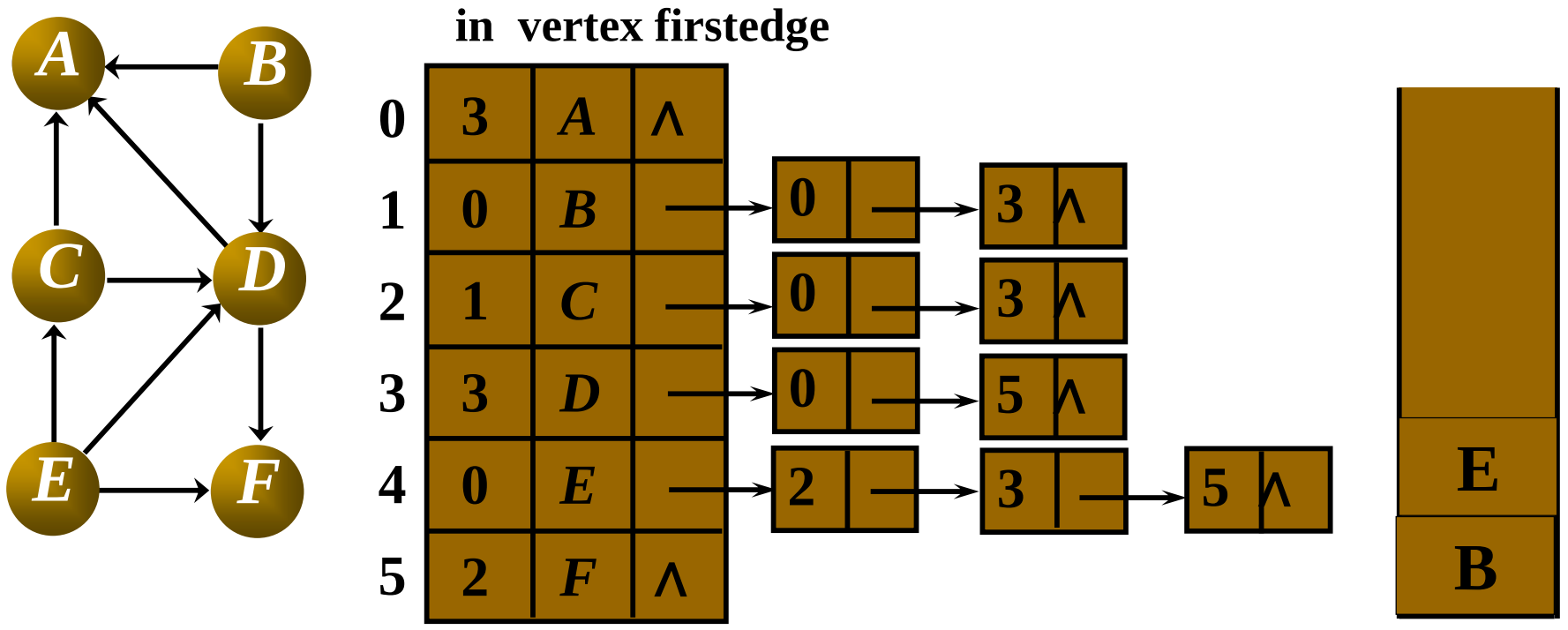
Data Structure

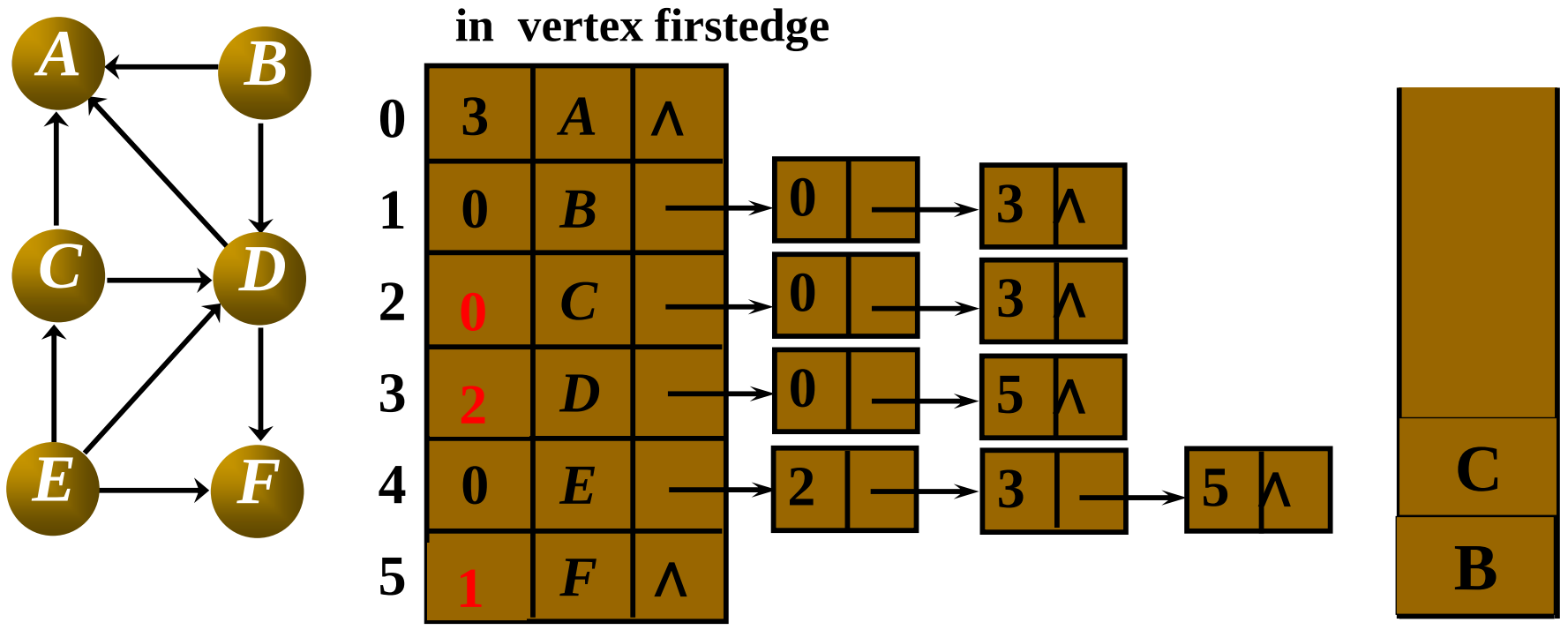
- Adjacent Lists with additional in-degree part

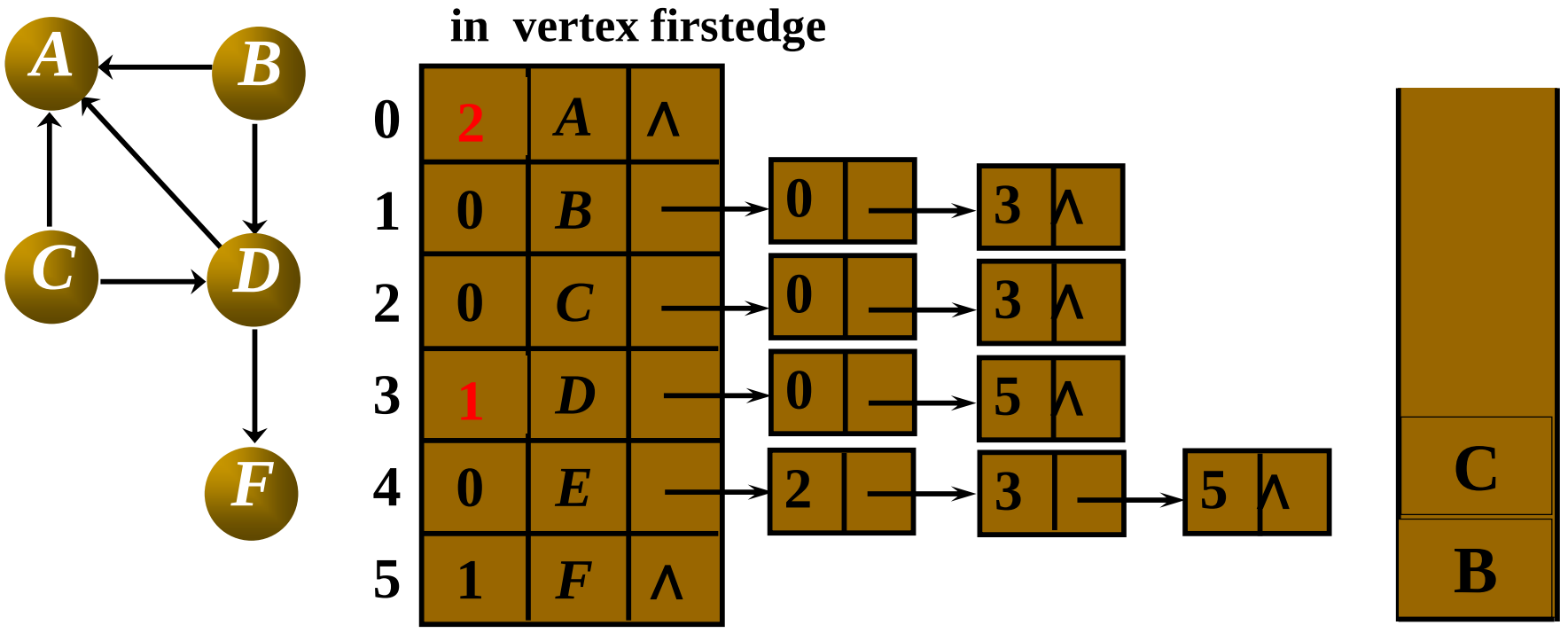
Vertex Node

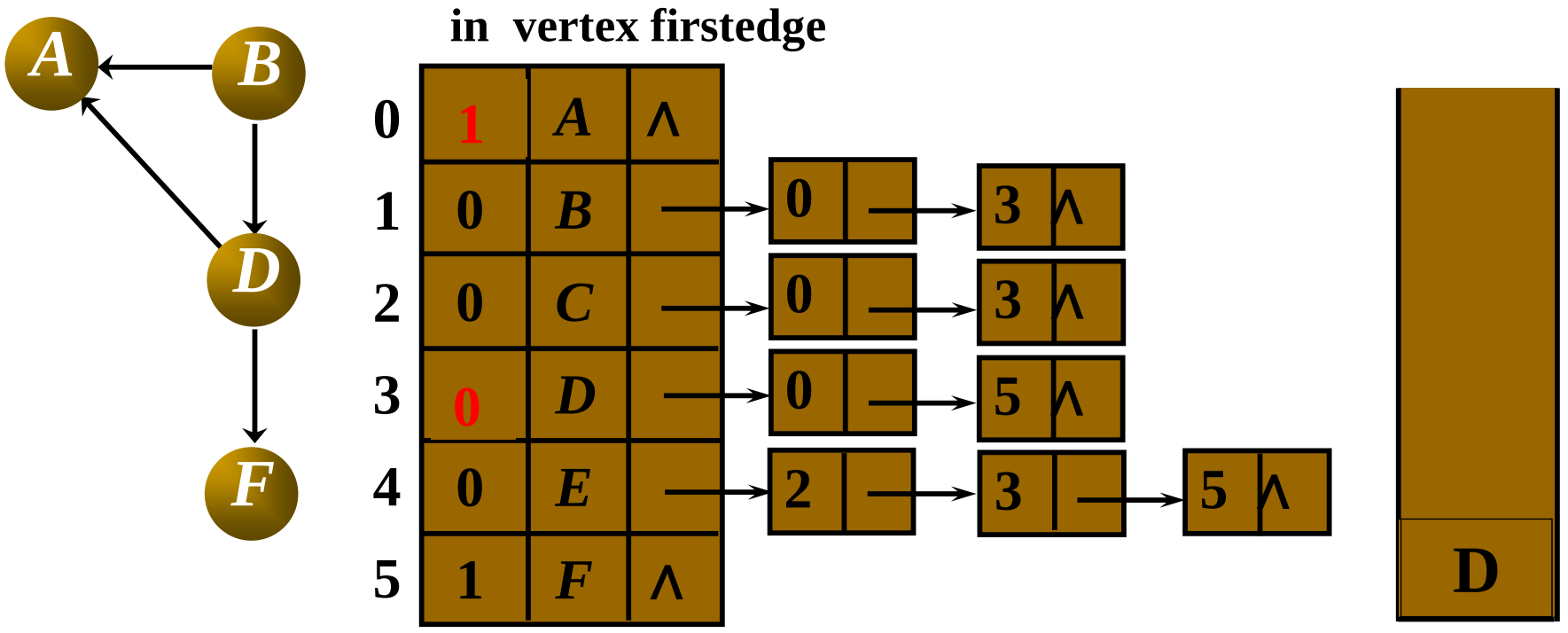
- Stack or Queue: store all no-precedent vertex

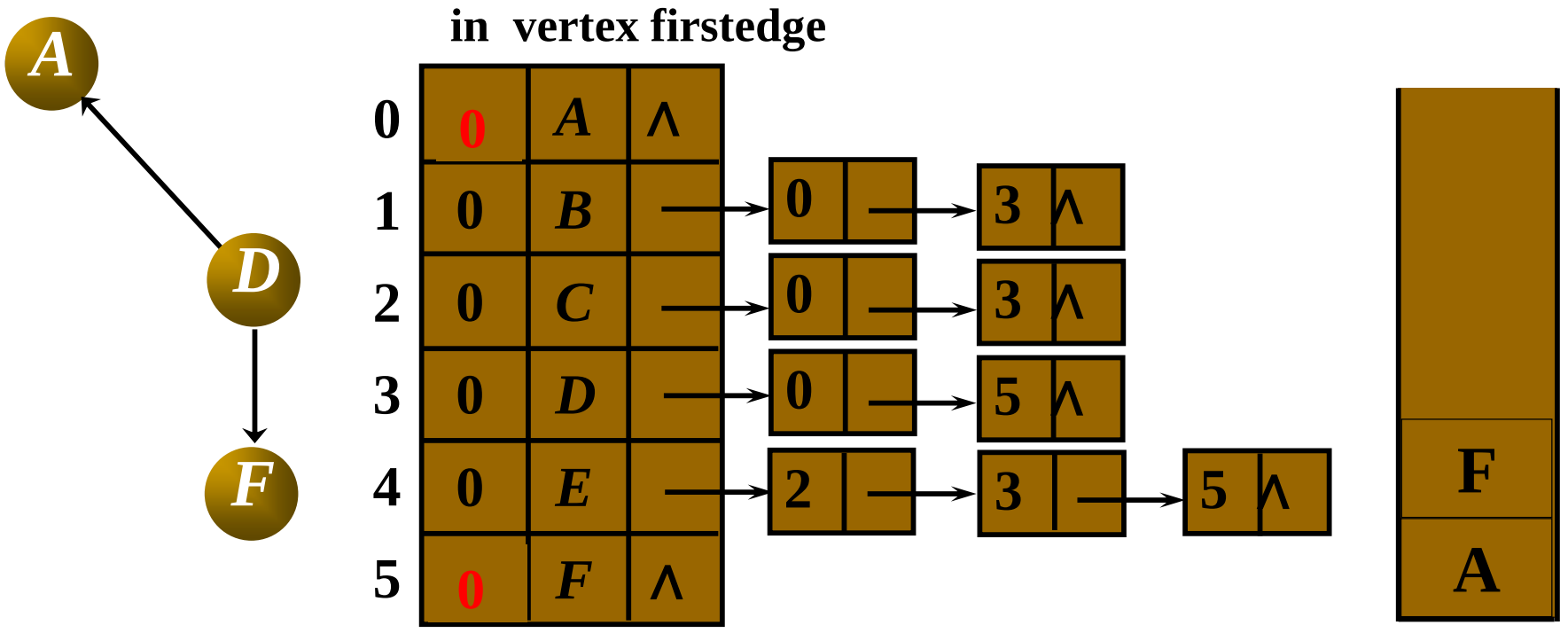

A
B
C
D
E
F
in vertex firstedge
0 3 A ∧
1 0 B
2 1 C
3 3 D
4 0 E
5 2 F
0
3 ∧
0
3 ∧
0
5 ∧
2
3
5 ∧

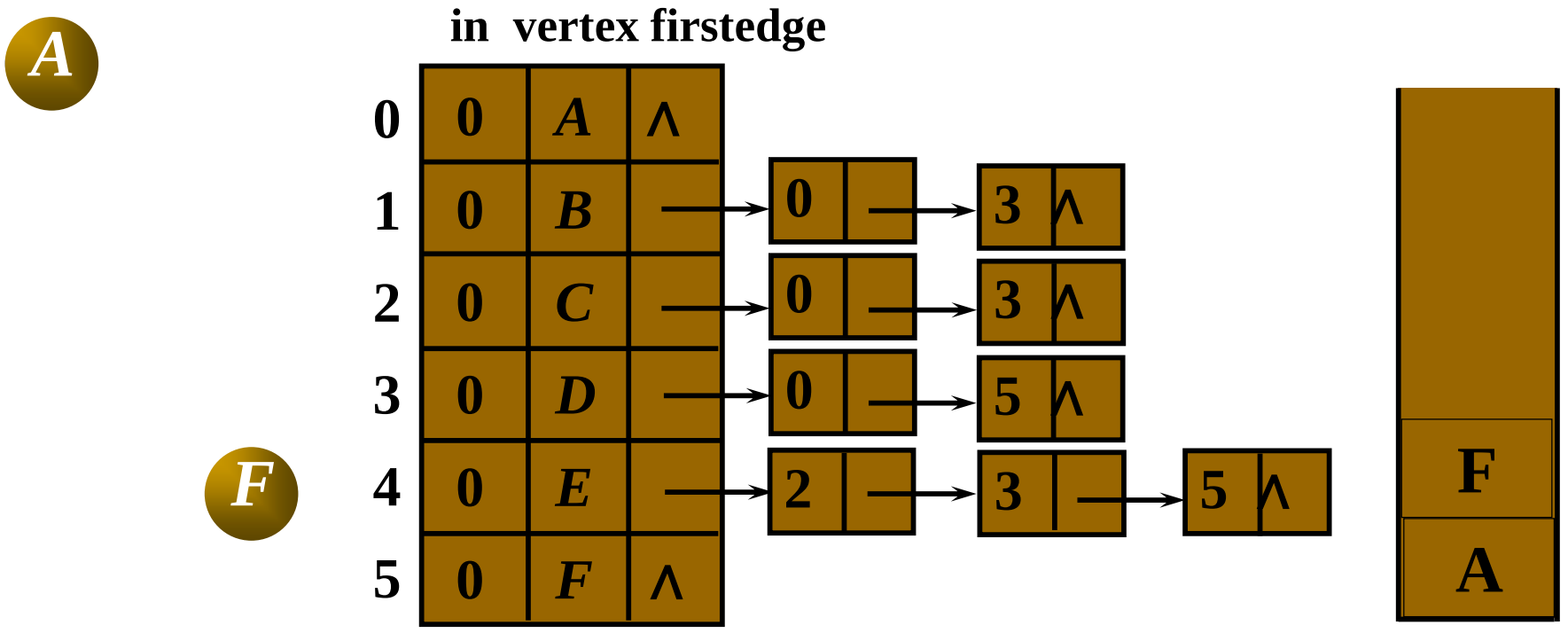

A
B
C
D
E
F
in vertex firstedge
0 3 A ∧
1 0 B
2 1 C
3 3 D
4 0 E
5 2 F ∧
0
3 ∧
0
3 ∧
0
5 ∧
2
3
5 ∧
E
B


A
B
C
D
E
F
in vertex firstedge
0 3 A ∧
1 0 B
2 0 C
3 2 D
4 0 E
5 1 F
0
3 ∧
0
3 ∧
0
5 ∧
2
3
5 ∧
C
B


A
B
C
D
F
in vertex firstedge
0 2 A ∧
1 0 B
2 0 C
3 1 D
4 0 E
5 1 F ∧
0 3 ∧
0 3 ∧
0 5 ∧
2 3 5 ∧
C
B


A
B
D
F
in vertex firstedge
0 1 A ∧
1 0 B
2 0 C
3 0 D
4 0 E
5 1 F ∧
0
3 ∧
0
3 ∧
0
5 ∧
2
3
5 ∧
D


A
D
F
in vertex firstedge
0 0 A ∧
1 0 B
2 0 C
3 0 D
4 0 E
5 0 F ∧
0 3 ∧
0 3 ∧
0 5 ∧
2 3 5 ∧
F
A


A
F
in vertex firstedge
0 0 A ∧
1 0 B
2 0 C
3 0 D
4 0 E
5 0 F ∧
0 3 ∧
0 3 ∧
0 5 ∧
2 3 5 ∧
F
A

Pseudo Code for Topological Sort

```
void topsort( graph G )
{
    QUEUE Q;
    int counter = 0;
    vertex v, w;
    Q = create_queue();
    for each vertex v
        if( indegree[v] = 0 )
            Q.add(v)
    while( !Q.isEmpty(  ) )
    {
      v =  Q.poll(); ++counter; /* assign next number */
      for each w adjacent to v
        if( --indegree[w] = 0 )
            enqueue( w, Q );
    }
     if( counter != NUM_VERTEX )
        error("Graph has a cycle");
}
```

Algorithm Performance

- The time to perform this algorithm is O(|E| + |V|) if adjacency lists are used. This is apparent when one realizes that the body of the for loop at line 8 is executed at most once per edge.
- The queue operations are done at most once per vertex, and the initialization steps also take time proportional to the size of the graph.